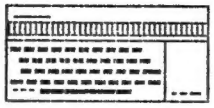



LIST

LONG ISLAND SINCLAIR TIMEX GROUP
INCORPORATING * NYTSE OF NEW YORK CITY
ISSUE: APRIL 1990

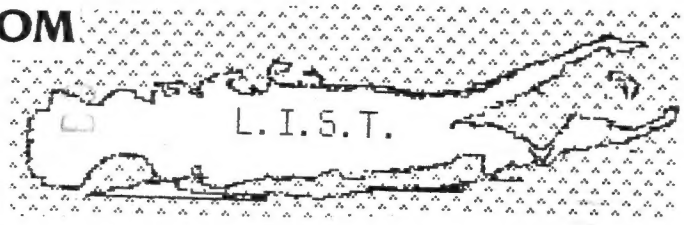
DISK DRIVES SUPPORTED:
1 OLIGER
2 LARKEN
3 AERCO

NEXT MEETING MAY 20

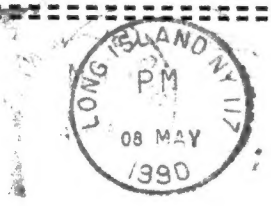
LIST MEMBERSHIP IS $15.00. LIBRARY TAPES ARE AVALIABLE, WRITE THE ADDRESS PRINTED BELOW.

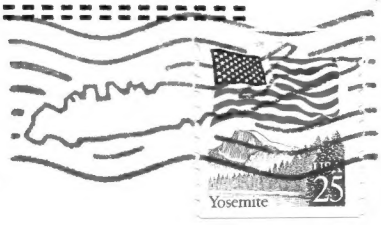

A Cheap and Simple EPROM Programmer

ZX81 Software



L.I.S.T.
5 PERI LANE
VALLEY STREAM, NY
11581





TO: Don Lambert JAN/91
3310 Clover Dr. S.W.
Cedar Rapids, IA
52404

FIRST CLASS MAIL
DATED MEETING NOTICE

UPPER RIGHT CORNER OF YOUR LABEL IS DATE OF LAST ISSUE.

LISTing

LIST OFFICERS

PRES.	HARVEY RAIT
TRES.	ROBERT MALLOY
REC.SEC.	STEVE KAYE
EDITOR.	FRED STERN
LIBR.	TOM SKAPINSKI

PLEASE SEND INQUIRIES TO:
LIST
MR. HARVEY RAIT
5 PERI LANE
VALLEY STREAM, N.Y. 11581
PLEASE SEND SUBMISSIONS TO:
LISTING
MR. FREDERIC STERN
214 ROBERTS ST.
HOLBROOK, N.Y. 11741

NYTSE

NYTSE MEETS THE MONDAY AFTER THE LIST MEETING AT;
MISS KIMS RESTAURANT
PARK AVENUE SOUTH
BETWEEN 21 ST. AND 22 ST.
MEETINGS START 7:30 PM.

COMING EVENTS:

MAY 20, 1990 LIST MEETING
MAY 21, 1990 NYTSE MEETING

MEETING MINUTES
APRIL 20,1990

THIS IS THE FIRST MEETING HELD AT THE WOODBURY NURSING HOME.

HARVEY CALLED THE MEETING TO ORDER AT 2:15 PM.

HARVEY AND FRED READ LETTERS RECEIVED SINCE THE LAST MEETING.
WE WELCOME 3 NEW MEMBERS;
WILLIE ROLDAN, BRONX N.Y.
MARK AND MARY YOST, CANTON CT.

HELP WANTED, HELP WANTED,

FRED STERN HAS BEEN ANSWERING LETTERS TO LIST AND EDITING THE NEWSLETTER. WE NEED A MEMBER TO TAKE OVER ANSWERING THE CORRESPONDENCE.
FRED WILL CONTINUE THE JOB UNTIL THE SUMMER BREAK.
INTERESTED MEMBERS CAN CONTACT EITHER FRED OR HARVEY.

NEWS, NEWS, NEWS, NEWS, NEWS,

WE RECEIVED A FLYER FROM BUDGET ROBOTICS, TUCSON AZ.BUDGET IS SELLING PARTS AND PERIPHERALS FOR THE ZX-81 AND TS1000. OF INTEREST ALSO IS AN AD FOR ZX-81 KITS THROUGH ZEBRA SYSTEMS FOR $49.95.

MAR/APR 90 SINCUS REPORTS THAT E. ARTHUR BROWN HAS AN AD ALSO FOR THE ZX-81 KIT FOR $39.95.

DEMONSTRATION

JIM DOUCAS AND BOB GILDER DEMONSTRATED NETWORKING USING 2 OR MORE @L COMPUTERS. JIM EXPLAINED THAT TWO PROGRAMS ARE REQUIRED TO GET THE SYSTEM GOING NETWORK ON TOOL KIT IS THE PRIME PROGRAM, BUT E-SERVE IS ALSO NEEDED TO CORRECT FAULTS IN THE NET PROGRAM.
I HOPE JIM OR BOB WRITE AN ARTICLE FOR LISTING TO BETTER EXPLAIN THE SOFTWARE AND HOOKUP.

COMING EVENTS

OUR NEXT MEETING WILL AGAIN BE HELD AT THE:
WOODBURY NURSING HOME
8533 JERICHO TURNPIKE
WOODBURY N.Y.

OUR JUNE 10, 1990 MEETING AND SWAPMEET WILL BE AT HARVEYS HOUSE IN VALLEY STREM.

SPECIAL THANK YOU

TO MS. KIM GIDIA AND THE STAFF OF THE WOODBURY NURSING HOME FOR THERE HOSPITALITY AND EFFERTS TO MAKE OUR MEETING A GREAT SUCCESS.

CLASSIFIEDS

THIS CLASSIFIED SECTION IS AVAILABLE TO ALL LIST MEMBERS FREE OF CHARGE.
THE ONLY RESTRICTION IS THAT IT IS TO BE USED ONLY FOR THE SEEKING, SELLING OR SWAPPING OF SINCLAIR, TIMEX OR MICRORACE COMPUTER EQUIPMENT, PERIPHERALS AND SOFTWARE.
LISTING, LIST, AND ITS OFFICERS DO NOT ENDORSE, WARRANTY, OR GUARANTEE ANY OF THE ITEMS LISTED IN THIS CLASSIFIED SECTION

IF YOU HAVE A COPY OF @-SAVE, FOR THE TS1000 PLEASE CONTACT FRED STERN 516-737-0963.

A FINAL WORD

MY NAME IS FRED STERN AND I AM THE EDITOR OF THIS EDITION OF LISTING.

THANK YOUS GO TO TOM SKAPINSKI FOR HIS ASSISTANCE.

ENCLOSED IN THIS ISSUE IS A PARTIAL LIST OF OUR MEMBERSHIP. NAMES AND ADDRESSES OF MANY MEMBERS WERE NOT INCLUDED BY THE MEMBERS REQUEST.

THE NOTICE OF THE KEN GORDON SHOW MENTIONED IN THE LAST NEWSLETTER WAS DELETD BEFORE GOING TO PRINT FOR OBVIOUS REASONS. SORRY FOR ANY INCONVENIENCES.
SEE YOU ALL AT THE NEXT MEETING.

LONG ISLAND SINCLAIR TIMEX GROUP

Isidore Goldsmith AUG/90
537 Monaco L, Kings Point
Delray Beach, FL
33446

(305)495-4318

Harvey Rait JAN/91
5 Peri Ln.
Valley Stream, NY
11581
(516) 791-6247

Bob Malloy JAN/91
412 Pacific St.
Massapequa Park, NY
11762

516-541-6731
Don Lambert JAN/91
3310 Clover Dr. S.W.
Cedar Rapids, IA
52404

Robert Gilder JAN/91
69 Jefferson Pl.
Massapequa, NY
11758

Jim Doucas JAN/90
1633 Crosby Rd.
Bronx, NY
10461

Hugo DiGiovanni JAN/90
6 Wallingford Dr.
Melville, NY
11747

Ike Walker JAN/91
253 Kenna Drive
So. Charleston W VA
25309-2646

304-346-2929

John Pazmino JAN/91
979 E 42 St.
Brooklyn, NY
11210

Stoney McMurray JAN/91
473 Westminster Rd
Brooklyn, NY
11218

(718)469-5948

Steve Kaye JAN/91
1281 E 86th St.
Brooklyn, NY
11236

Myles Cohen JAN/91
10 E 95 St.
New York, NY
10128

Ken Diederich NOV/90
312 N. Bailey
Jacksonville, AR
72076

Thomas Skapinski JAN/91
7 Atkinson Lane
Coram, NY
11727

(516)732-1825

Fred Stern JAN/91
214 Roberts St.
Holbrook, NY
11741
(516) 737-0963

Philip Florio JAN/91
28-35 Jordan St.
Bayside, NY
11358

Ed Lee JAN/91
17 w. 17th St.
New York, NY
10011

Joeseph F. LaPunzina NOV/90
91 Bay 31 St
Brooklyn, NY
11214

Thomas W. Jennens MAR/91
33901 Lawton Ave.
Eastlake, OH
44095

John D. Solomon Jr. JAN/91
139 North Terrace Ave.
Mount Vernon, NY
10550

Charles J. Musumerci MAR/90
Coler Memorial Hosp. Ward A-42
Roosevelt Island, NY
10044

Don & Eleanor Lamen MAY/90
R.D. #3 Box 3404
Windsor, NY
13865

Mark Scheck JUN/90
86 Holmes St.
Statford, CT
06497

(203)378-4186

Charles R. Byler SEP/90
P.O. Box 5061
APO SF, CA
96519

Jay Siegel MAR/91
5213 9th Ave.
Brooklyn, NY
11220

(718)436-5367

Armand R. Martin OCT/90
431 Dubuc Street
Winnepeg, Manitoba
CANADA R2H 1G4

John J. Rossi III OCT/90
6 Beech Ct.
College Point, NY
11356

Jess Wyder DEC/90
17 Academy St.
Fishkill, NY
12524-1301

Nicholas I. Oshana,Jr. NOV/90
101 Treble Rd.
Bristol, CT
06010

Dr. Armand Drucker JAN/91
194-02A 67th Ave.
Fresh Meadows, NY
11356

Dr. Raymond Kaufman FEB/91
3755 Henry Hudson Pky. Apt 4F
Bronx, NY
10463

Richard K.Norek FEB/91
188 St. Felix Ave
Cheektowaga, NY
14227-1228

Robert M. Curnutt FEB/91
10400 Truxton Rd.
Adelphi, MD
20783

Paul W. Chomitz JAN/91
95 Delancey St.
New York, NY
10002

L.I.S.T.

L.I.S.T.

TRY THIS

A Cheap and Simple EPROM Programmer

Dan Schaaf

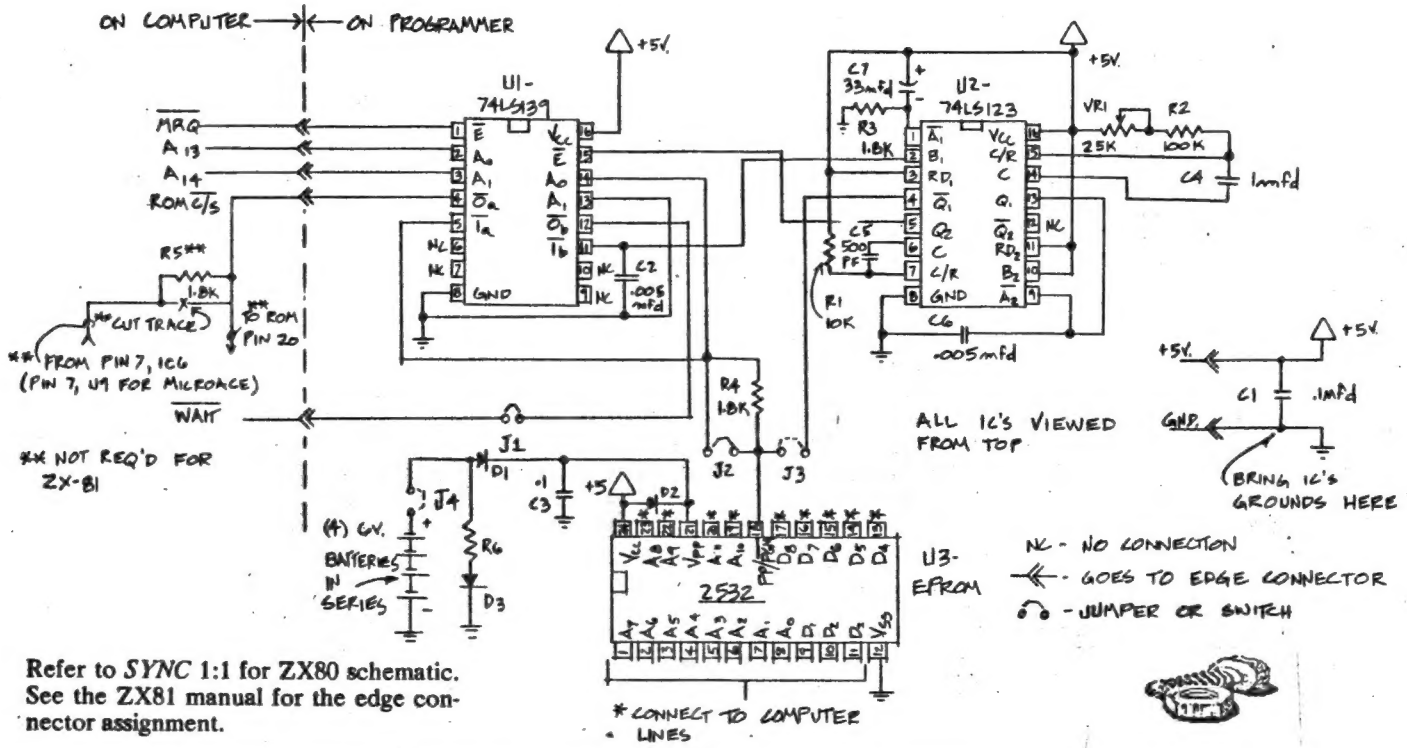
"What more could I want?" That's what you usually say before you buy something. But you learn after you have had it for a while that there is a lot more to be wanted. Past issues of *SYNC* have had several articles to remedy heart-felt deficiencies in the Sinclair Basics: SCROLLs, PAUSEs, READs, DATAs, string handling routines, etc.

This article will show you how to build an add-on device which will allow you to have any of these routines, plus any you can design yourself, permanently and instantly available to you in one 4K EPROM without the need to LOAD those routines from tape into the machine (a waste of time and of RAM). Those interested in peripherals for the Sinclairs (floppy disk, electronic typewriters, etc.) can design monitors and controllers for those devices. Eventually you can program independent microprocessors for special purposes. An EPROM programmer can become a real liberator of the power inherent in the Z80.

4K EPROMs now cost less than $10.00. 4K was chosen because of its cost advantage and simplicity of programming. 4K is a good healthy chunk of memory as witnessed by the

Dan Schaaf, 306 N. Carroll, Michigan City, IN 46360.

Figure 1. Schematic Diagram.



Refer to *SYNC* 1:1 for ZX80 schematic. See the ZX81 manual for the edge connector assignment.

Figure 2. Timing Diagram.

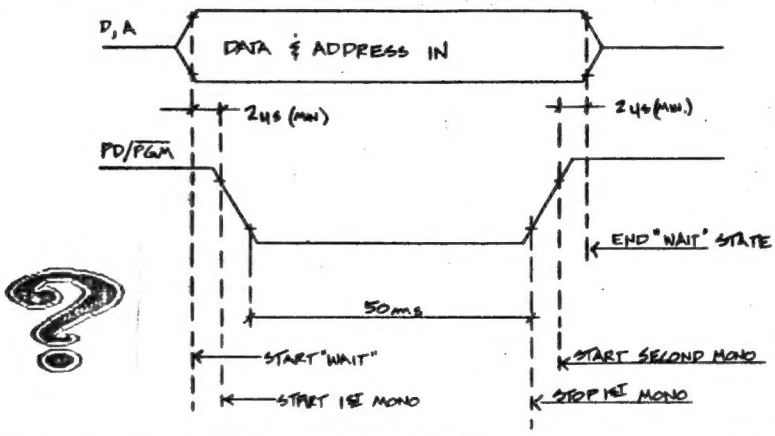
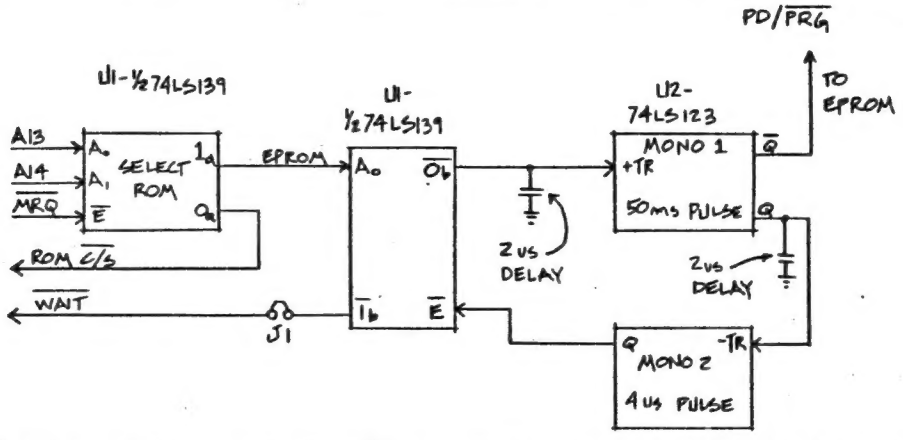


Figure 3. Block Diagram.



The Block Diagram shows how the basic EPROM timing needs are met by the circuit. As soon as a Memory Request is made into the EPROM, a Wait state is initiated. This holds data and addresses on the bus line until cleared by the timing loop. The beginning of the 50 ms pulse to the PD/PGM pin is delayed 2 μs (capacitor C2). 2 μs (capacitor C6) after the 50 ms pulse ends, a 4 μs pulse from the second mono clears the Wait state and allows the computer's bus lines to operate as normal.

Sinclair 4K Basic. Although not necessary since EPROMs come "erased" from the factory, it is useful to have or have access to an ultraviolet lamp for erasing the EPROM. They can be bought new for $70.00, but some can be found at flea markets and are sometimes known as rock hound lights used to fluoresce organic and other materials in rocks. At $70.00, it should be noted, however, seven fresh EPROMs could be bought.

Once built, programming proceeds by POKEing data into the proper addresses. USR is then used if the data was a machine language program, or PEEK if the data was a table, and also PEEK to check that programming worked properly.

What Is an EPROM?

EPROM stands for Erasable Programmable Read Only Memory. An EPROM has some of the same characteristics as the ROM in your machine that contains the Basic language:

1) Data may be read from the EPROM once an address has been presented to the address bus.

2) Data is retained when the power is turned off, almost forever.

There are differences: EPROMs need to be programmed a byte at a time; ROMs are made in large batches each with the same program on them. EPROMs can be erased (for reprogramming) by the use of ultraviolet radiation; ROMs can never be erased. In a sense, an EPROM is midway betway a ROM and a RAM. Typically an EPROM has the same gross dimensions as a ROM and the same pin-out as a ROM (since the computer uses both the same way) but in addition it has a little quartz window on top that allows the ultraviolet light into the EPROM for erasing. Erasing results in all data being lost at all addresses. There is no selective erasing.

How Does an EPROM Work?

The basic element of the EPROM is a capacitor which holds a charge. The estimated life time (time constant) of the charge is 7 to 10 years. Each bit is represented within the EPROM by one capacitor. A sensing amplifier converts this static charge to a TTL level voltage that appears on the data lines as either a "1" or "0". Ultraviolet radiation has sufficient energy to dislodge electrons on the capacitor surface, and as these photoelectrons are generated the capacitor will discharge and data will be erased. Most EPROMs read a discharged capacitor as a "1" and a charged capacitor as a "0"'. 25 volts are used to charge the 32,768 tiny capacitors on the chip. A typical ultraviolet lamp takes about half an hour to discharge the EPROM. Manufacturers advise that a tape cover be put over the window to prevent accidental erasure.

How the EPROM Programmer Works

Looking at the data sheets for the EPROM selected, one sees various requirements:

1) Data and address information must be at TTL compatable voltages.

2) For programming, 25 volts must be applied to pin 21 (V_{pp}).

3) Data and addresses must be held constant for 50 milli-seconds.

4) A certain timing sequence must be carried out with regards to the activation of the programming pin, pin 18 (See Timing Diagram.)

1) and 2) are easily met. The Z80 has TTL compatable voltage levels and four 6 volt batteries will provide 25 volts. 3) and 4) are harder to meet. A POKE command only holds address and data information on the busses for about 550 nanoseconds, almost 100,000 times too short for proper EPROM programming.

The solution is to use the $\overline{\text{WAIT}}$ pin on the Z80. When the $\overline{\text{WAIT}}$ pin is active (low voltage) all bus information is held constant until the pin voltage goes high again. The circuit shown has a one-shot whose time period is set to 50 microseconds. The other timing requirements are taken care of by the circuit as well, as explained next.

When the EPROM is addressed (a proper combination of A13, A14, and $\overline{\text{MRQ}}$, memory request) by the Z80, U1 switches the internal ROM off (via $\overline{O}_a$-pin 4) and turns on the EPROM (via $\overline{I}_a$-pin 5). If J1 is in, a $\overline{\text{WAIT}}$ state is started when $\overline{O}_b$ pin 12 of U1 goes low. At the same time, pin 11 $\overline{I}_b$ of U1 goes high initiating the 50 millisecond one-shot of U2 (VR1 + R2, C4). Q1 of U2 goes low for 50 milliseconds as does $\overline{\text{PGM}}$, pin 18 of U3, which then programs the EPROM. 2 milliseconds after the end of the 50 millisecond programming period, the second one-shot of U2 (R1, C5) disables the Enable pin, pin 15, of U1 thus clearing the $\overline{\text{WAIT}}$ state and allowing the computer to process the next instruction.

The design was such as to use a minimum of parts. Certain problems therefore exist which can be avoided by being aware of them. The main concern is that a PEEK command, or USR command into the EPROM when the EPROM is in the "program" mode will write useless data into the EPROM. Therefore, *do not* PEEK or USR into the EPROM without disconnecting the 25 volts (J4 out).

If you are familiar with dynamic RAM specifications (the type of RAM used in the 16K-64K expansion units), you will know that they specify a maximum refresh period usually of 2 milliseconds. During a WAIT cycle there is no refreshing. So if the data sheets are to be believed, dynamic RAMs should not work with this EPROMer. However, I have found that dynamic RAMs can tolerate as much as one or two seconds without refresh and without loss of data. I have used my EPROM programmer with 16K and the newer 64K dynamic RAMs and have had no troubles. If you run into trouble, disconnect the dynamic RAMs and use the itnernal static RAMs, which need no refresh.

Making It

The construction method described here is the one I used and may not be best for you.

What I usually do is lay out the ICs (or IC sockets) on a board, making sure that there is enough room for miscellaneous components and that the wiring which will interface with the computer is conveniently close to the edge of the board. I seek to minimize runs, especially power runs (5 and 25 volt). I then put in the other components and use the leads as much as possible to wire the circuit. Every step of the way I check and double check that my wiring corresponds to the schematic. ICs get confusing since you must look at them from the top and the bottom. (See schematic.)

Many years ago I got a lot of color-coded telephone wire (what more could I want?) from an abandoned factory in my home town.

Program 1.

```
10 FOR L=0 TO 399
20 POKE 16000,0
30 NEXT L
```

Program 2.

```
10 FOR L=0 TO 4191
20 IF PEEK (L+12288)<255 THEN
PRINT L,PEEK (L+12288)
30 NEXT L
```

Program 3.

```
 5 IF INKEY$<>"P" THEN GOTO 5
10 POKE 16383,65
20 SAVE "P"
30 GOTO 20
```

Program 4.

```
SAME AS PROGRAM 2
```

Program 5.

```
 5 IF NOT INKEY$="P" THEN GOTO
5
10 POKE M,N1
20 POKE M+1,N2
30 POKE M+2,N3
 .   (REST OF MACHINE LANGUAGE
 .     PROGRAM)
 .
 .
1000 SAVE "P"
1010 GOTO 1000
```

Note: M = first address at which the data is to be put. N1, etc. is the data.

I use that to wire the rest of the circuit. I recheck the circuit against the schematic and then solder all the points. After that I use a Volt-Ohm Meter to test the continuity of each run as well as that two runs have not accidently shorted. Good, clean, logical construction habits save hours of needless trouble shooting later. I try to document what and how I have wired it after I am done.

So now you have a board wired as in the schematic. Next it must be interfaced with the computer. Here again I used a method which may not be right for you. Any method is acceptable, including hardwiring it with color coded telephone wire. However, this could limit future expansion. AP Products (as shown on page 42 of the 1982 Jameco Catalog) has jumper headers and cable assemblies which can be used. I used two straight double row, 40 post jumper headers. One set I soldered onto the EPROM board. The other set I soldered onto the computer's edge connector lower forty.

I then interconnected the two boards with a 6" AP flat ribbon cable. Long runs of data lines can cause problems and should be avoided. Of the four remaining edge connector positions, only two are needed: D_7 and 5+ volts. These I hard wired with a connector in between for disconnecting.

The EPROM is a MOS device and is susceptible to damage due to static discharge. I have not been overly cautious, yet I have never had trouble with static charge destroying a

Figure 4. Parts List.

Part.		Radio Shack Number. (If more than one of item in package, then number is listed only the number of times needed.)
	ICs	
U1	74LS139	** (See below)
U2	74Ls123	**
U3	TMS 2532, 450 ns.	**
	Resistors	
R1	10K	271-034
R2	100K	271-045
R3	1.8K	271-1324
R4	1.8K	
R5	1.8K	
R6	3.3, ½ watt (optional)	271-028
	Capacitors	
C1	.1 mfd	272-1069
C2	.005 mfd	272-130
C3	.1 mfd	
C4	1 mfd	272-1419
C5	470 pfd	272-125
C6	.005 mfd	
C7	33 mfd, 16V.	272-1426
	Diodes	
D1	Diode	276-1101
D2	Diode	
D3	LED (optional)	Various available
	Variable Resistors	
VR1	25K	
	Miscellaneous	
J1, J4	Jumpers or one way switches	Any suitable
J2-3,	3-post jumper or 2-way switch	Any suitable
PC Board		276-153
IC socket for U3 - 24 pin*		276-1989
Batteries	(4) - 6 volt lantern	Eveready 509 or equal
Connector Cable	(See article "Making It")	

* The other IC's can be soldered into place without too much fear of damage or having to replace them. But the EPROM, and any MOS device, really should be in a socket.

** Available through many mail order houses, such as Jameco, Active Electronics, JDR Microdevices, etc., Addresses and prices can be found in the back of most hobby electronics magazines.

MOS device. One should, however, make sure before handling the EPROM that you are discharged to the ground of the computer board. Do not place EPROM into socket until both the timing and the 25v circuit have been checked out.

In making the EPROM programmer I ran into difficulties. The main one was that the EPROM pin (address 7) was defective and only one half of the memory was addressable. At first I assumed that my circuit was in error. After several days I convinced myself that the EPROM was bad and returned it for exchange and bought another chip as well. The exchanged chip had the same problem while the new chip appears to be working fine. (I also accidently applied 25v to the 5v supply which fried 2 RAMS but in the process fortunately spared the rest of the computer.)

Trouble Shooting Before Putting the EPROM In

In *SYNC* (2:1) I showed a method of monitoring computer activities by injecting signals into the video output. This method is useful in dynamically testing the board and connection to the computer. With the computer on and the board connected, note the different patterns that the pins on the Z80 make relevant to the EPROM. For instance, the patterns for A_0 at the edge connector should be the same as A_0 at the EPROM.

Program 6. A Machine Language Program to be POKEd into the EPROM.

```
          LD HL  (E-file)      ; Load HL with E-file location the end
                                  of the variables list.  A string must
                                  be at the end of that list.
          EX HL,DE
          LD H, B              ;Clear HL for addition
          LD L, B
NEXT:     DEC DE
          LD A,(DE)            ;See if it's a variable name
          CMP , 129
          JRC -NEXT            ;If not check next byte
AGAIN:    INC DE
          LD A, (DE)
          CMP , Ø1             ;If quote token (Ø1) then you're done and
          RETZ                 ;you'll return to the BASIC program with
          ADD HL               ;the number in HL.  USR returns HL to the
          ADD HL               ;program.
          ADD HL               ;Shift HL four places: 16 times.
          ADD HL
          SUB, 28              ;Convert letter to number
          JRC, +ERROR          ;Error if string token less than Ø
          CMP, 16              ;See if token is greater than "f"
          JRC +NO ERROR        ;If not, jump to NO ERROR
ERROR:    RST Ø8 , 9           ;ERROR ends BASIC program with ERROR code "A"
NO ERROR: LD C,A
          ADD HL,BC            ;Add converted token to HL
          JR AGAIN             ;Do again
```

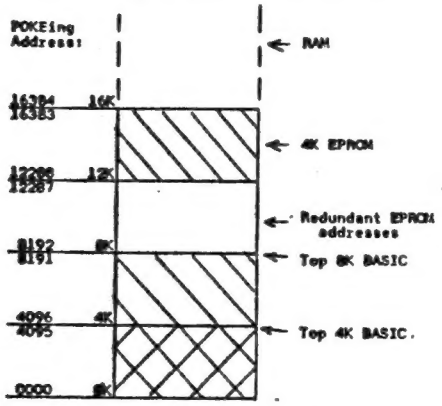
Decimal listing for Program 6.

```
 0-42    1-10    2-64    3-235   4-96
 5-104   6-27    7-26    8-254   9-129
10-56   11-250  12-19   13-26   14-254
15-01   16-200  17-41   18-41   19-41
20-41   21-214  22-28   23-56   24-4
25-254  26-16   27-56   28-2    29-207
30-9    31-79   32-9    33-24   34-233
```

Chart 1.

MODE:	J1	J2	J3	J4
Test	in	out	in	out
Program	in	out	in	in
Read	out	in	out	out

Chart 2. Memory Locations.



The microprocessor starts at zero and proceeds up in addresses, doing each instruction in turn until it eventually loops back to a lower address.

Repeat for A_0 to A_{11} and D_0 to D_7 as well. Make sure that there are five volts where there should be on the board. If all looks good, then you are ready to check the timing circuit.

Program 1 will allow you to set VR1 for timing the one-shot. First, put the programmer into the test mode (see Chart 1). Then run the program. It should take as close to 20 seconds to run as you can measure. Adjust VR1 until the program takes 20 seconds. If you cannot adjust the programmer to 20 seconds, then R may be changed, lower value to make it faster, higher value to slow it down.

Applying the 25 Volts

The 8K user should read the next section before going further.

Since the 25v can be so damaging, triple check that the 25v line only runs to pin 21 of the EPROM and that the diode is in properly and works properly. Then turn the power off.

Put the EPROM in, turn the power back on, and RUN Program 2. If Program 2 ran badly, go back and check the wiring. If it ran okay, then RUN Program 3. (4K ROM apply 25v before RUNning the program; 8K ROM, during the run.) After the program is over and the 25v are removed, run Program 4. Program 4 checks to see that your wiring has not created redundant addressing in the memory. If the program prints more than one address, then most likely an address line is open (or the EPROM has a defective address pin), and you must go back to the previous section and dynamically check the address and data lines. If program ran okay, then you are ready to program the EPROM with real data.

Verify that when the 25v are connected to the Vpp that only 3 to 30 ma are drawn by that pin. Also, if some other 25v source is used, make sure it is well regulated to within + 1v of 25v.

Programming the EPROM

Programming is simple. Put the Eprom programmer into the program mode (Chart 1) and then POKE data into the EPROM addresses. Because of the way the 8K ROM uses the Refresh cycle, I have had trouble on occasion while the video is on. Therefore, for the 8K ROM the video must be "off" while the EPROM programmer is in the program mode. Program 5 is used for the 8K ROM. Do not have the 25v on when the 8K ROM video is

Program 7. Entering Hexadecimal Data.

```
10 FOR L=0 TO 19
20 PRINT L;"-";
30 INPUT I0
40 PRINT I0
50 POKE L+M,USR N
60 NEXT L
```

Note: M = first address to which data is to be POKEd. N = address of Program 6 in the EPROM. This program *cannot* be used to POKE into the EPROM.

on, or else bad data may be written into the EPROM. Line 5 of the program is a loop which waits for the "P" key to be pressed. While the program is waiting, you must set the EPROM programmer to the "program" mode. Then press "P" and the program will POKE the data into the EPROM. You will know that the programming is done when the screen starts to gitter with garbage as the program gets SAVEd in line 1000. Now place the EPROMer into the "read" mode. Then press BREAK.

Two sample programs are shown which work for the 4K Basic. One converts a hexidecimal string into a number (Program 6). The other program converts a number into a hexidecimal representation which can be PRINTed (Program 8). Programs 7 and 9 show their uses in Basic programs.

Putting It to Work

This construction project is really an open ended affair, just as computers are. Once operating, your limits are your imagination, patience, and money. Though not particularly difficult to build, like all projects it will take fortitude to keep on going when things look hopeless (and they will).

Published programs should be checked to see if any changes must be made when relocating them to the EPROM addresses. Relative jumps and calls to inside the program have to be changed. An example of one particular problem is illustrated by trying to adapt the "Scroll" routine of Logan (*SYNC* 1:4). The routine requires three bytes of RAM, one to pass the "Leave" parameter, the other two for an internal program register. The addresses of the RAM bytes were immediately before the subroutine. Obviously, an EPROM can not be used for RAM. My solution was to pass the "Leave" parameter through the RANDOMISE command and use the Syntax pointer location for the INTERNAL PROGRAM register since no Syntax pointing is required during the subroutine.

There have been many good subroutines in *SYNC* and the general techniques outlined in such articles can be used as long as they are not used blindly but understood and modified as needed. A good book on Z80 machine language programming will help a great deal.

A log of the addresses should be kept so that you know where what is and that you do not over write onto existing programs. Also tape copies of all programs should be kept in case it becomes necessary to reprogram the whole EPROM.

Program 8. A Machine Language Program to be POKEd into the EPROM.

```
LD HL,(VARS)          ;Load HL with location of first Variable.
                      ;Variable must be there from BASIC
INC HL                ;program and must be a one letter name.
LD A, (HL)            ;Get past variable name.
LD B, A
AND A, ØF h           ;Mask last four bits
ADD A, 28             ;Convert to number token
LD HL, (E-file)       ;Load HL with the address of the end of
                      ;the variable list.  A two character
                      ;string must be there from the BASIC
                      ;program.
DEC HL
DEC HL                ;Move past quote in string to
DEC HL                ;last charactor in string
LD (HL), A            ;Load converted token into string
DEC HL                ;Move pointer to first charactor in string.
LD B, A               ;Get rest of number.
AND A,FØ h            ;Mask upper four bits.
RR
RR                    ;Shift upper bits into lower bits
RR
RR
ADD A, 28             ;Convert to number token.
LD(HL) , A            ;Place in upper charactor of string.
LD HL,ØØØØ            ;Clear HL just for neatness' sake.
RET                   ;Return to BASIC with the last string
                      ;being a hexidecimal representation
                      ;of the number in the first variable.
                      ;Note:  Only numbers Ø to 255 are converted.
```

I have run my programmer with the power supply provided with the computer. However, when I started to add other devices, it quickly became necessary to supplement the power supply. If your system is already taxing the power supply, you will have to consider expanding it.

Construction Outline

1) Gather parts into one place and fit them into the PC board. Use the IC socket for the EPROM.

2) Solder the circuit together.

3) Test the board for proper connections.

4) Check and adjust the timing resistor. Check the 25v line.

5) Place the EPROM in and test it with the various programs (2 through 4).

6) If it is okay, program in some simple routines and see if they work.

Decimal listing for Program 8.

0-42	1-8	2-64	3-35	4-126
5-71	6-230	7-15	8-198	9-28
10-42	11-10	12-64	13-43	14-43
15-43	16-119	17-43	18-120	19-230
20-240	21-15	22-15	23-15	24-15
25-198	26-28	27-119	28-33	29-Ø
30-Ø	31-201			

A Cheap and Simple EPROM Programmer (3:1)

The author asks to have added at the end of the section "How the EPROM Programmer Works," p. 80, the following paragraph:

As designed, the EPROM programmer will not work with 64K memory modules. The reason is that they generate their own ROM C/S signals which would conflict with those generated by the programmer. However, once programmed, the EPROM could be placed into a ROM socket, provided that the pin configuration is compatible or is made compatible. The problem here is that no one EPROM pin assignment scheme exists. To make a "standard" ROM socket compatible with the EPROM, the following modifications to the ROM circuit (not the programmer) are required: traces to pins 18 and 20 should be reversed. The trace to pin 21 should be cut and pin 21 should be tied to Vcc (+5V). (Note: I have reversed the usual assignments of pins 19 and 20 to aid in compatibility.) In all probability, the ROM circuit provided would require such modifications.

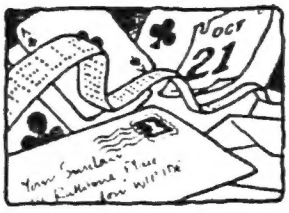
Program 9. Display of Data as Hexadecimal Numbers.

```
 5 LET X=0
10 FOR L=0 TO 19
20 LET X=PEEK (L+M)
30 LET I$="XX"
40 LET X=USR N
50 PRINT L;"-";I$,
60 NEXT L
```

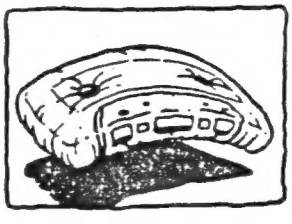
Note: Line 5 reserves the first variable in the variable list for the subroutine. This must be done before the FOR command sets up its variable. The FOR variable cannot be used for the first variable, nor can variables with more than one character in the name. Line 30 makes sure that there is a two character string at the end of the variables list. M = first address of the area to be displayed. N = address of Program 8 in the EPROM.

ZX81 Software

```
  1 REM ZX-81/TS1000/TS1500
  2 SLOW
  3 REM [NEWER] PRESS RUN,BUT USE
  GOTO 1 INSTEAD
  5 REM V.3,PARTS.B3,PARTSDATA-
BASE,COPYRIGHT BY BILL HARMER,
OTTAWA,1986,PLACED IN PUBLIC DOM
AIN 1990
 35 LET T$="PARTS DATABASE"
 55 REM GO TO STARTING DISPLAY
 60 GOSUB 9600
 65 REM GO TO INTRO INFO
 70 GOSUB 9850
 80 CLS
100 GOSUB 9800
110 PRINT "                    [MENU]"
115 PRINT
120 PRINT "1-START A NEW DATABA
SE"
125 PRINT
130 PRINT "2-ADD TO AN EXISTING
 DATABASE"
135 PRINT
140 PRINT "3-SCAN DATA"
145 PRINT
150 PRINT "4-SAVE PROGRAM WITH
DATA"
155 PRINT
160 PRINT "5-QUIT PROGRAM"
165 PRINT
186 PRINT "6-LIST DATA"
195 PRINT AT 20,0;"[ENTER] THE NU
MBER OF YOUR CHOICE"
200 INPUT MENU
210 CLS
220 IF MENU=1 THEN LET S$="$"
250 GOTO MENU*1000
300 GOTO 80
970 CLS
990 PRINT "THATS ALL FOLKS"
992 PRINT
995 PRINT "PROGRAM USED=";PEEK
16404+256*PEEK 16405-16384;" BYT
ES"
999 STOP
1005 LET P$=""
1010 GOSUB 9800
1015 PRINT AT 20,0;"ENTER ""QUIT
"" TO GO TO MENU"
1020 PRINT AT 4,0;"[ENTER] PARTS N
UMBER-MAX 12 DIGITS"
1025 PRINT AT 21,0;"
                    "
1026 IF P$<>"" THEN PRINT AT 17,
0;"PREVIOUS PART NO.";P$
1030 INPUT N$
1035 IF N$="QUIT" THEN GOTO 1400
1040 IF LEN N$>12 THEN GOSUB 110
0
1045 IF LEN N$>12 THEN GOTO 1030
1050 IF LEN N$<12 THEN GOSUB 120
0
1055 PRINT AT 20,0;"
                    "
1060 PRINT AT 4,0;"PARTS NUMBER=
";N$;"
                    "
1065 PRINT AT 21,0;"
                    "
1070 PRINT AT 6,0;"[ENTER] QUANTIT
Y (UP TO 999)"
1080 INPUT Q$
1082 IF LEN Q$=2 THEN LET Q$="0"
+Q$
1083 IF LEN Q$=1 THEN LET Q$="00
"+Q$
1085 IF VAL Q$>999 OR VAL Q$<0 T
HEN PRINT AT 21,0;"*UNACCEPTABLE
 QUANTITY-RE-ENTER*"
1090 IF VAL Q$>999 OR VAL Q$<0 T
HEN GOTO 1080
1091 PRINT AT 6,0;"QUANTITY=";Q$
;"
                    "
1092 PRINT AT 20,0;"
                    "
1093 GOSUB 9500
1094 LET S$=S$+N$+Q$+"$"
1095 LET P$=N$
1096 GOTO 1010
1100 REM TOO LONG PART NUMBER
1130 PRINT AT 21,0;"*PART NUMBER
 TOO LONG*RE-ENTER* "
1140 PRINT AT 4,0;"NO. ENTERED="
;N$;" IS TOO LONG"
1160 RETURN
1200 REM SUB TO LENGTHEN PARTS
NO. FIELD
1205 LET M$=""
1210 LET L=LEN N$
1220 FOR X=1 TO 12-L
1230 LET M$=M$+" "
1240 NEXT X
1250 LET N$=M$+N$
1260 RETURN
1400 GOTO 255
2000 REM ADD TO EXISTING DATABAS
E
2010 GOTO 1000
3000 REM DATA SCAN ROUTINE
3005 PRINT AT 17,0;"
                    "
3010 GOSUB 9800
3015 IF S$="$" THEN PRINT "NO DA
TA IN FILE"
3020 LET E=((LEN S$-1)/16)
3030 PRINT AT 19,0;"[PRESS] ""0""
TO GO TO MENU,""A"" TO REPEAT,"
"S"" TO STOP,""C"" TO CONT-  INU
E,""D""TO DELETE FILE"
3035 LET C=1
3040 FOR X=1 TO E
3042 PRINT AT 17,0;"
                    "
3045 PRINT AT 4,0;"FILE NO. ";C;
"
                    "
3050 LET K=((X-1)*16)+1
3055 IF S$(K+16)="*" THEN GOTO 3
150
3056 LET C=C+1
3060 PRINT AT 6,0;"PART NUMBER="
;S$(K+1 TO K+12)
3070 PRINT AT 8,0;"QUANTITY=";S$
(K+13 TO K+15)
3080 FOR Z=1 TO 35
3090 IF INKEY$="S" THEN GOSUB 33
00
3100 IF INKEY$="0" THEN GOTO 320
0
3110 NEXT Z
3150 NEXT X
3155 PRINT AT 17,0;"FILE END REA
CHED"
3160 IF INKEY$="0" THEN GOTO 320
0
3170 IF INKEY$="A" THEN GOTO 300
0
3180 GOTO 3160
3200 GOTO 255
3300 REM CONTROL SUBROUTINE
3310 IF INKEY$="C" THEN GOTO 333
0
3312 IF INKEY$="0" THEN GOTO 320
0
3313 IF INKEY$="A" THEN GOTO 300
0
3314 IF INKEY$="D" THEN LET S$(K
+16)="*"
3317 IF S$(K+16)="*" THEN PRINT
AT 17,0;"FILE DELETED"
3320 GOTO 3310
3330 RETURN
4000 REM SAVE  PROGRAM WITH DATA
4040 CLS
4050 PRINT AT 9,4;"[PRESS]-[SAVE
]-[ENTER]-[RUNNING]"
4060 PRINT AT 15,2;"[START-CASSET
TE-RECORDER-NOW]"
4070 PAUSE VAL "200"
```




```
4080 CLS
4090 SAVE "PARTS DB"
4095 GOTO 1
5000 REM QUIT PROGRAM
5010 GOTO 970
6000 REM LIST ROUTINE
6005 LET L=15
6010 IF S$="$" THEN PRINT "NO DA
TA IN FILE"
6020 IF S$="$" THEN STOP
6025 REM CALCULATE NUMBER OF
PARTS NOS.
6030 LET E=((LEN S$-1)/16)
6035 IF E<16 THEN LET N=1
6036 IF E<16 THEN LET L=INT E
6040 IF E<16 THEN GOTO 6055
6050 FOR N=1 TO (INT (E/16)+1)
6055 CLS
6056 GOSUB 9800
6057 PRINT "PART NUMBER","QUANTI
TY"
6060 FOR Z=1 TO L
6070 LET X=((N-1)*15)+Z
6080 LET K=((X-1)*16)+1
6085 IF K+16>LEN S$ THEN GOTO 61
35
6090 IF S$(K+16)="*" THEN GOTO 6
135
6130 PRINT S$(K+1 TO K+12),S$(K+
13 TO K+15)
6135 IF Z=L AND N=(INT (E/16)+1)
 THEN GOTO 6170
6140 NEXT Z
6145 GOSUB 9500
6150 IF INT E<16 THEN GOTO 6170
6160 NEXT N
6170 PRINT "      NO MORE ITEMS"
6180 GOSUB 9500
6190 GOTO 255
9000 REM PROGRAM SAVE ROUTINE
9010 CLS
9015 CLEAR
9020 PRINT AT 9,4;"CASSETTE SAVE
#BEGINNING"
9025 PRINT AT 15,2;"START CASSET
TE RECORDER NOW"
9030 PAUSE VAL "200"
9035 CLS
9040 SAVE "PARTS DB"
9050 GOTO 20
9299 STOP
9500 REM FLASH TO GO ON SUBR.
9505 PRINT AT 21,22;"  "
9510 PRINT AT 21,0;"PRESS ANY KE
Y TO GO ON"
9520 FOR X=1 TO 3
9530 IF INKEY$<>"" THEN GOTO 958
0
9535 NEXT X
9540 PRINT AT 21,0;"PRESS"
9550 FOR X=1 TO 3
9555 IF INKEY$<>"" THEN GOTO 958
0
9560 NEXT X
9570 GOTO 9510
9580 CLS
9590 RETURN
9600 REM INITIAL DISPLAY
9630 PRINT AT 7,0;"
                    "
9640 PRINT AT 8,0;"
                    "
9650 IF LEN T$>28 THEN PRINT "CH
ANGE TITLE PRINT LINES 9650-9700
)"
9655 PRINT "
";
9660 PRINT AT 9,INT ((30-LEN T$)
/2);T$
9670 PRINT AT 9,31;"
"
9720 PRINT AT 10,0;"
                    "
9730 PRINT AT 11,0;"
                    "
9740 PRINT AT 15,5;"PROGRAM (REV
,1990) BY
BILL HARMER,
OTTAWA"
9760 PRINT AT 21,0;"PRESS ""ENTE
R"" TO GO ON"
9765 INPUT A$
9770 CLS
9780 RETURN
9800 REM TITLE PRINT SUBROUTINE
9810 PRINT AT 0,INT ((32-LEN T$)
/2);T$
9815 PRINT
9820 RETURN
9850 REM PRELIMINARY INFO HERE
9860 GOSUB 9800
9870 PRINT "PARTS DATABASE IS A
SIMPLE DATA-BASE WHICH WILL HOLD
 A PARTS     NUMBER UP TO 12 DIGI
TS LONG AND A 3 DIGIT QUANTITY F
OR EACH PART"
9875 PRINT
9880 PRINT "THIS DATABASE MAY PR
OVE USEFUL  FOR PARTS INVENTORY
OR AS A     BUILDING BLOCK TO AD
D TO TO MAKEA MORE SOPHISTICATED
 DATABASE"
9885 PRINT
9890 PRINT "THIS DATABASE VERSIO
N DOES NOT  HAVE A SORT OR SEARC
H FACILITY  BUT DOES HAVE A RECO
RD DELETE   ABILITY"
9960 GOSUB 9500
9990 RETURN
9991 REM NOTES
9997 REM SUBROUTINES:9600 START-
ING DISPLAY   9500 FLASH TO GO
ON    9800 PAGE TOP TITLE
9055 TITLE REPEAT PAGE
```



LARKEN SYSTEM

DISK DRIVE MENU LOADER

EASY TO USE AND CONVEIENT

Just place the highlighted bar on the program name you wish to LOAD. This program uses the CAT feature to assemble the list of the programs to load. It can handle 100 titles, and you can go back and forth over the entire list of names.

The program was written for L.I.S.T. (Long Island Sinclair Timex) group by: Keith Skapinski and is available on a 5 1/4 in. disk if you prefer. Order from Tom Skapinski, 7 Atkinson Lane. Coram, NY 11727 (Cost $5.00)

INSTRUCTIONS:
Enter the first listing then compile using Timachine by Novelsoft. Save as directed by Timachine. Name "DISKME.CO" then reload at 50000, and save as "DISKME.C1",CODE 50000,3745 Enter the second listing, the basic/loader. Run: then break and goto 200 to save the two parts. I hope you enjoy using this program. Keep Computing!

This prog. utilizes CODE written by ?

```
  5 REM !USR 50000
 10 REM ! OPEN #
 30 CLS
 50 FOR i=VAL "63000" TO VAL "6
3077"
 60 READ a: POKE i,a: NEXT i
 70 DATA VAL "195",VAL "43",VAL
 "246",VAL "195",VAL "72",VAL "2
48",VAL "195",VAL "104",VAL "248
",VAL "243"
 80 DATA 205,93,0,VAL "201",VAL
 "58",VAL "100",0,VAL "251",VAL
"201",VAL "205"
 90 DATA VAL "33",VAL "246",VAL
 "58",VAL "176",VAL "92",VAL "50
",VAL "29",VAL "32",VAL "205",VA
L "126"
 100 DATA 0,VAL "205",VAL "123",
0,VAL "33",VAL "112",VAL "32",VA
L "17",VAL "156",VAL "224"
 110>DATA 1,0,VAL "20",VAL "237"
,VAL "176",VAL "195",VAL "38",VA
L "246",VAL "205",VAL "33"
 120 DATA VAL "246",VAL "58",VAL
 "176",VAL "92",VAL "50",VAL "29
",VAL "32",VAL "33",VAL "156",VA
L "224"
 130 DATA VAL "17",VAL "112",VAL
 "32",1,0,VAL "20",VAL "237",VAL
 "176",VAL "205",VAL "150"
 140 DATA 0,VAL "205",VAL "126",
0,VAL "205",VAL "120",0,VAL "195
",VAL "28",VAL "246"
 200 REM PROGRAM
 210 POKE VAL "23728",0: LET o=0
: BORDER o: PAPER o: INK 7: DIM
a$(VAL "100",VAL "10"): DIM t(VA
L "100")
 220>CLS : PRINT INVERSE 1;"
    DISK MENU PROGRAM          "; I
NVERSE 0;"
 230 RANDOMIZE USR VAL "63000"
 235 LET name=VAL "57688"
 240 FOR i=PI/PI TO VAL "100"
 250 IF PEEK (name+VAL "3")=o TH
EN GO TO 300
 260 IF PEEK (name+VAL "3")=VAL
"254" THEN LET name=name+34: GO
TO 250
 270 FOR j=PI/PI TO VAL "9": LET
 a$(i,j)=CHR$ PEEK (name+j): NEX
T j
 275 LET t(i)=PEEK (name+11)
 280 LET name=name+34
 290 NEXT i
 300 FOR i=VAL "1" TO VAL "100":
 IF a$(i,VAL "1")=" " THEN GO TO
 VAL "320"
 310 NEXT i
 320 LET p=i-VAL "1"
 510 PRINT #PI/PI;AT o,o; INVERS
E PI/PI;" <SPACE>=MOVE CURSOR <EN
TER>=LOAD<7>=PG UP  <8>=BREAK  <
6>=PG DOWN"
 520 LET x=p/VAL "20": LET x=VAL
 "1" AND x>0 AND x<=1)+(2 AND x>
1 AND x<=2)+(3 AND x>2 AND x<=3)
+(4 AND x>3 AND x<=4)+(5 AND x>4
)
 530 LET c=PI/PI
 535 LET d=2
 540 PRINT AT 2,0;: IF x>c THEN
FOR i=VAL "(c*20)-19" TO VAL "c*
20"
 550 IF x=c THEN FOR i=VAL "(c*2
0)-19" TO p
 560 PRINT TAB VAL "11";a$(i),
 570 NEXT i
 580 IF x=c THEN FOR i=1 TO ((c*
20)-p): PRINT ,,: NEXT i
 590 PRINT AT d,11; OVER 1; INVE
RSE 1;"  "
 600 PAUSE 0: LET q$=INKEY$: IF
q$="" THEN GO TO 600
 602 IF 0$="B" OR 0$="b" THEN ST
OP
 605 IF q$="  " THEN PRINT AT d,1
1; OVER 1; INVERSE 1;"
 610 IF q$="  " THEN LET d=d+1
 620 IF d=22 THEN LET d=2
 622 IF c=x AND d=22-((c*20)-p)
THEN LET d=2
 625 IF q$="  " THEN GO TO 590
 630 IF q$="6" THEN LET c=c+1
 640 IF c>x THEN LET c=x: BEEP .
01,10: GO TO 600
 650 IF q$="7" THEN LET c=c-1
 660 IF c=0 THEN LET c=1: BEEP .
01,10: GO TO 600
 665 IF q$="6" OR q$="7" THEN LE
T d=VAL "2"
 670 IF q$=CHR$ 13 THEN GO TO 69
0
 680 GO TO 540
 690 LET l$=a$(((c*20)-19)+d-2)
 695 FOR i=1 TO 10: IF l$(i)="  "
 THEN LET l$=l$( TO i-1): GO TO
700
 697 NEXT i
 700 CLS : PRINT "LOADING ";l$
 705 FOR I=65368 TO 65367+LEN l$
: POKE I,CODE l$(I-65367): NEXT
I: POKE I,CODE ":"
 740 REM ! CLOSE #
9000 CLEAR : PRINT #4: SAVE "DIS
KME.BT" LINE 1
```

```
 10 REM DISK MENU LOADER
 18 RANDOMIZE USR 100: OPEN #4,
"dd"
 20 CLS
 30 PRINT #4: LOAD "DISKME.C1"C
ODE 50000
 40 RANDOMIZE USR 50000
 50 IF PEEK 65368=0 THEN GO TO
120
 60 LET l$="": LET l=65368
 70 IF PEEK l=42 THEN GO TO 90
 80 LET l$=l$+CHR$ PEEK l: LET
l=l+1: GO TO 70
 90 IF l$(LEN l$-1)="C" THEN PR
INT #4: LOAD l$CODE : STOP
 100 PRINT #4: LOAD l$
 110 STOP
 120 CLS
 130 PRINT "B=Break program and
go to BASIC C=Change current dri
ve number"
 140 PAUSE 0: LET q$=INKEY$: IF
q$="" THEN GO TO 140
 150 IF q$="B" OR q$="b" THEN ST
OP
 160 IF q$<>"c" AND q$<>"C" THEN
 GO TO 140
 170 CLS : PRINT "Which drive (0
-3)?"
 180 INPUT d
 190 PRINT #4: GO TO d: GO TO 40
 200 REM SAVE
 210 PRINT #4: SAVE "DISKME.B1"
LINE 10
 220 PRINT #4: SAVE "DISKME.C1"C
ODE 50000,3745
```